PAPER • OPEN ACCESS

Comparison of memory thresholds for planar qudit geometries

To cite this article: Jacob Marks *et al* 2017 *New J. Phys.* **19** 113022

New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG
IOP Institute of Physics

Published in partnership with: Deutsche Physikalische Gesellschaft and the Institute of Physics

OPEN ACCESS

RECEIVED
20 February 2017

REVISED
11 August 2017

ACCEPTED FOR PUBLICATION
16 October 2017

PUBLISHED
15 November 2017

PAPER

Comparison of memory thresholds for planar qudit geometries

Jacob Marks[1,2,6], **Tomas Jochym-O'Connor**[2,3,4] **and Vlad Gheorghiu**[2,5]

[1] Department of Physics, Yale University, New Haven, CT 06520, United States of America
[2] Institute for Quantum Computing, University of Waterloo, Waterloo, ON N2L 3G1, Canada
[3] Department of Physics and Astronomy, University of Waterloo, Waterloo, ON N2L 3G1, Canada
[4] Walter Burke Institute for Theoretical Physics, Institute for Quantum Information and Matter, California Institute of Technology, Pasadena, CA 91125, United States of America
[5] Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON N2L 3G1, Canada
[6] Author to whom all correspondence should be addressed.

E-mail: jacob.marks@yale.edu, tjoc@caltech.edu **and** vlad.gheorghiu@uwaterloo.ca

Keywords: quantum, error correction, color codes, qudit, fault-tolerant

Abstract

We introduce and analyze a new type of decoding algorithm called general color clustering, based on renormalization group methods, to be used in qudit color codes. The performance of this decoder is analyzed under a generalized bit-flip error model, and is used to obtain the first memory threshold estimates for qudit 6-6-6 color codes. The proposed decoder is compared with similar decoding schemes for qudit surface codes as well as the current leading qubit decoders for both sets of codes. We find that, as with surface codes, clustering performs sub-optimally for qubit color codes, giving a threshold of 5.6% compared to the 8.0% obtained through surface projection decoding methods. However, the threshold rate increases by up to 112% for large qudit dimensions, plateauing around 11.9%. All the analysis is performed using QTop, a new open-source software for simulating and visualizing topological quantum error correcting codes.

1. Introduction

Quantum error correction (QEC) is of paramount importance for quantum information processing schemes, as any implementation will have imperfections that can lead to loss of coherence. Quantum error correcting codes (QECC), first introduced by Shor [1] two decades ago, seek to address such imperfections in the hopes of ensuring global protection. These codes generalize and extend the notion of classical error correction to both bit and phase flip Pauli errors to protect 'fragile' quantum states against undesired noise [1–4]. Since their inception, a multitude of techniques for constructing good QECC have been developed, such as CSS codes [2, 3], stabilizer codes [4], cluster-state based codes [5].

For a physical architecture to implement arbitrary quantum algorithms, it must be able to suppress potential errors that could affect the physical system. Fault-tolerant quantum computation allows for the scalable correction of errors in a controllable manner, and is characterized by the presence of a threshold theorem [6–8]. Namely, given a QECC, provided that the physical error rate of the gates is below a certain *threshold*, the logical error rate can be made arbitrarily small by extending the code distance. However, computing the threshold for a given code is computationally challenging. Moreover, the value of the threshold will depend on the type of decoder used in the code: once a code develops errors, a classical decoding algorithm must be used to return the code to the codespace. The best decoders are fast and result in the fewest logical errors.

Topological codes, introduced by Kitaev [9], are a subclass of stabilizer codes which make use of topological features in order to protect against local physical errors. These are also among the leading candidates for experimental fault-tolerant implementations. Perhaps the most well-known instance of a topological code is the *surface code*, in which data qubits (two-level quantum systems) are placed on a square lattice and the error correction is performed via measuring appropriate stabilizer generators on a shifted 'dual' lattice. Another type of topological code, the *color code* [10], is produced by tiling a surface with three-colorable faces, and associating

a distinct variety of stabilizer with each color (usually red, green, and blue). These color codes combine the topological error-protection of the surface code with transversal implementations of Clifford gates, allowing for increased ease in logical computation. While most research thus far has focused on qubits, both surface and color codes can be generalized to D-level quantum systems, or *qudits*, which can take on linear superpositions of D distinct values. Indeed, early numerical results suggest that qudits may give better performance by providing more information about the specific set of errors that has occurred [11–13].

The first contribution of our work is QTop [14], a universal numerical framework for simulation and visualization of topological codes of arbitrary code distance, and qudit dimension. Our software includes surface codes, and 6-6-6 color codes—one of the most experimentally-promising semi-regular tilings of the plane, and allows for simulation under arbitrary noise models. Our framework is modular, object-oriented, and simple to use. It can be used to test new decoders, and extending it to 3D color codes and more exotic topological systems is straightforward.

The second main contribution is a decoder for qudit color codes. This decoder is inspired by renormalization group clustering techniques prevalent in the qudit surface code decoding literature. We implement the proposed decoder in QTop, and analyze its performance under the generalized bit-flip memory noise model, as the code is a CSS code that can address bit and phase flip errors independently. Memory noise, or errors introduced on the physical data qudits alone (while measurement circuits are perfect), is typically used as a first estimate for the viability of an error correcting code for fault-tolerant computation. We obtain a threshold value of 5.6% in the case of the qubit color code, a drop from 8.0% we obtain using the surface projection method. This type of drop is expected due to the inherently approximate nature of renormalization decoders, and is seen in the case of surface code renormalization decoders [11]. Moreover, as in the case of the surface code, the threshold rate increases with qudit dimension, saturating at a value that is above that of the idealized qubit case, that is 11.9%.

The remainder of this paper is organized as follows. In section 2 we motivate our study of topological codes, and provide a detailed overview of the two most prevalent types—surface and color codes. In section 3 we formulate the problem of decoding. Section 4 describes our novel method for decoding qudit color codes, and in section 5 we present threshold results for such codes under various error models. Finally in section 6 we summarize our results, raise some open questions and discuss possible future research directions.

2. Topological codes

2.1. Motivation

Topological QECC are regarded as highly promising schemes for fault-tolerant quantum computation as logical states are encoded in highly non-local degrees of freedom of the system. Therefore, in order for physical errors to lead to logical faults, error chains will have to form that will be as non-local as the logical states, unlikely in the event that the noise is not strongly correlated. Moreover, topological codes are characterized by stabilizers that are typically of low weight[7], and more importantly, local. Therefore, local errors will lead to local excitations in the stabilizer space, typically allowing for efficient decoding algorithms. In this section, we describe two of the most studied classes of topological codes for the purposes of quantum computing, the surface code and the color code.

2.2. Surface codes: benefits and limitations

The surface code was among the first class of topological codes, proposed by Kitaev [9]. The surface code is a special instance of the Toric code structure, where smooth and rough boundaries are introduced allowing for the storage of a single logical qubit. Namely, complementary sets of anti-commuting logical operators are represented by excitations that connect differing sets of boundary types, satisfying anti-commutation by intersecting at an odd number of sites in the lattice.

One of the primary advantages of the surface code is that the stabilizers are given by weight-4 operators, allowing for high threshold values in the case of circuit level noise. However, the surface code is limited in its set of logical transversal gates, that is the set of gates that can be implemented in a bit-wise fashion throughout the code. As such, the surface code faces potential increases in overhead for the implementation of logical gates.

2.3. Color codes: topology and transversality

Color codes were first introduced as an alternative two-dimensional geometrical architecture to the surface code [10]. The color code construction is characterized by having three different boundary types, unlike the surface code, and logical operators are formed by connecting all three boundaries. In this sense, the color code has a

[7] The weight of a tensor product of Pauli operators is defined as the number of non-identity Pauli operators in the product.

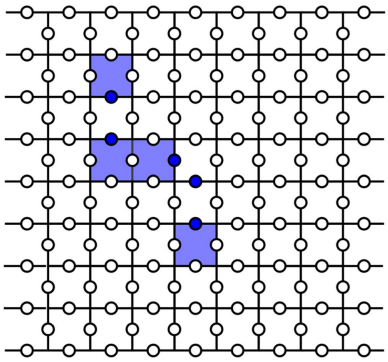

Figure 1. Example of a distance 9 implementation of the surface code, where Z stabilizers are defined by weight-4 plaquette operators, while X stabilizers are given by weight-4 star operators. In the presented example, an X error has occurred on the blue colored qubits, causing excitations at the colored plaquettes. These colored plaquettes will serve as the inputs to the decoding algorithm.

symmetry between the boundaries that is not present in the surface code. This symmetry manifests itself in the transversal operators of the code, which we detail below.

Color codes are CSS codes whose stabilizers, in planar geometries, are given by plaquette operators only. Since all plaquettes hold both X and Z type operators, all plaquettes must intersect at an even number of qubits in order to satisfy commutativity of the stabilizers. The graph is therefore 3-valent and 3-colorable, thus giving rise to the notion of color in the code setting [10, 15]. This condition on the structure of the graph leads to a restricted set of possible configurations for the stabilizers of the code, resulting in possible stabilizer weight distributions of 6-6-6, 4-8-8, and 4-6-12 [16]. Unlike the surface code, there is a symmetry in the two types of stabilizers in the code, resulting in a transversal Hadamard gate. Moreover, the generalized phase gate can be implemented transversally by choosing a careful distribution of positive and negative rotations. As such, all Clifford gates can be implemented transversally for the color code[8], a significant advantage over the surface code. These properties of the color code have allowed recent research in fault-tolerance to develop schemes for alternative methods to magic-state distillation for universal quantum computation [17–19]. In addition, a recent result by Bombin has shown that three-dimensional color codes can be used in single-shot QEC to circumvent the need to store historical syndrome data when correcting for errors in measurement [20].

3. The problem of decoding

3.1. Minimum weight perfect matching

In topological codes, error syndromes are registered as excitations of stabilizers on the topological surface in question. That is, given an error chain composed of a particular Pauli operator, the resulting syndromes whose measurement results will change will be the endpoints of the error chain. We denote such changes in the expectation value of the syndromes as excitations. Treating flipped syndromes as excitations is rooted in the fact that the endpoints of these error chains behave like anyonic particles, with associated braiding statistics depending on the type of error. As such, any decoder will need a notion of the topology at hand. The code distance will refer to the weight of the logical Pauli operator with smallest support. between different excitations will allow to minimize the probability of introducing a logical error upon decoding. In figure 1, we provide an example of a weight-4 X error on a distance 9 surface code patch. Given that the error is of low weight, proper decoding should lead to its correction. The inputs to the decoding algorithm are most simply described in the dual lattice picture, as seen in figures 2 and 3, where lattice sites correspond to syndrome locations, and edges correspond to shared qubits between pairs of stabilizers.

The input to the decoding algorithm is the locations of the excitations, or flagged syndromes, and the corresponding distance between pairs of excitations. The distance between excitations is given by the minimal number of qubits connecting a chain with the excitations at the endpoints. The simplest form of decoder—the Greedy algorithm—leads to poor performance. Given a look-up table of all the excitation pairs and their corresponding distances, the Greedy decoding algorithm finds the smallest weight pair, and assigns a correction along the connecting chain. Then, these two excitations are removed from the table, and the next-smallest pair is found, and so forth until all pairs of excitations have been corrected. In the example provided in figure 1, the two neighboring excitations are first matched up together (since they are distance 1 apart), and then the remaining

[8] The CNOT gate is also transversal since the code is a CSS code.

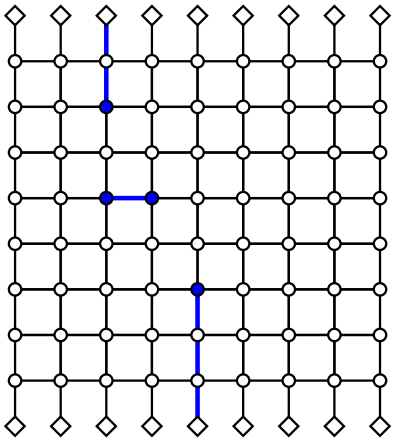

Figure 2. Greedy implementation of decoding for the error presented in figure 1. Note that overall the weight of the corrected error chain is higher, even though the two nearest excitations have been paired together.

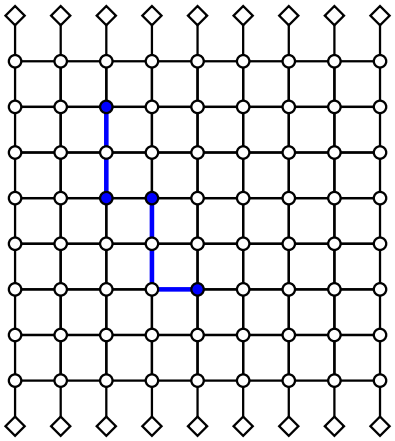

Figure 3. Implementation of the min-weight perfect matching decoder for the error presented in figure 1. The decoder optimizes to find smaller weight corrections than the Greedy decoder by searching for pairs of higher weight.

excitations are matched up through the boundary (as this forms the minimal-weight error between these excitations). Unfortunately, when combined with the initial errors in the code, the remaining error will now form a logical error since the Greedy decoder accidentally paired syndromes that were quite delocalized.

A more accurate form of correction can be achieved using an algorithm called min-weight perfect matching (MWPM). The MWPM algorithm is based on Edmonds' blossom algorithm [21], where excitations are matched together by considering growing sets of matchings of different weights. Moreover, the algorithm is efficient in the number of input pairs, scaling roughly as N^6, where N is the total number of excitations[9]. In the example shown in figure 3, the MWPM algorithm matches together two pairs of excitations that are of distance 2 and 3. Therefore, the total distance of the corresponding pairs is 5, as opposed to in the case of the Greedy decoder, where the total correction was of weight $1 + 5 = 6$. The MWPM algorithm aims to minimize over this sum.

3.2. Qudit codes

Qubit stabilizer codes can be generalized to the case of *qudits* through the introduction of generalized Pauli operators, the Heisenberg–Weyl operators. A qudit resides in a D-dimensional Hilbert space, and the generalized Pauli operators take the form:

[9] The blossom algorithm scales as P^3, where P is the total number of pairs. Since there are N excitations, there are on the order of N^2 pairs of excitations.

$$X = \sum_{j=0}^{D-1} |j \oplus 1\rangle\langle j|, \qquad Z = \sum_{j=0}^{D-1} \omega^j |j\rangle\langle j|, \tag{1}$$

where addition is taken mod D and $\omega := e^{2\pi i/D}$ is the Dth complex root of unity. Intuitively, the X and Z operators generalize the regular Pauli operators, and while no longer being self-inverse, do satisfy $X^D = Z^D = I$. Stabilizer codes will again be defined as the '+1' eigenspace of the generalized Pauli operators. Importantly, the operators no longer just simply commute or anti-commute, but rather have a generalized form of commutation relationship, $X^k Z^l = \omega^{-kl} Z^l X^k$. Therefore, when generalizing topological CSS codes to the setting of qudits, it will no longer be sufficient for X and Z type stabilizers to just overlap at an even number of locations, as the phases will no longer automatically cancel out. In order to overcome this restriction, particular orderings for each of the operators are chosen in order to satisfy commutativity of the stabilizers [9]. We will expand upon this notion further in upcoming sections. One distinct advantage of qudit codes is that due to the enlarged local Hilbert spaces, stabilizer measurements can return a much larger set of possibilities. As such, increased information about chains of errors can be detected by neighboring plaquettes, since a weight-2 error on a given stabilizer will not necessarily lead to a trivial syndrome measurement. This phenomenon has been shown to produce higher thresholds for qudit codes, in comparison with similar qubit codes, when decoding takes this additional information into account [11].

Emphasizing this increased difficulty in decoding, the MWPM decoding technique described in the previous section translates to hyper-graph matching in the case of qudit decoding on the surface code, a computationally inefficient task. As such, modified decoding techniques have to be used in the case of qudit codes, and these are typically based on a form of renormalization [11]. The process involves iteratively increasing the scale, at each scale identifying maximally connected disjoint clusters, and if they are neutral (the sum of excitations $= 0$ mod D), annihilating them through charge transport, as described in more detail in a later section.

Just as in the case of qubit codes, topological qudit codes exhibit thresholds due to the percolation of errors. That is, given a particular fixed topological code and decoder, for independent errors at a rate below a certain threshold, the code will be corrected with high accuracy, reducing the logical error rate of the state with respect to the physical error rate. However, once the statistical fluctuations of errors become too great, larger error regions will form, leading to high likelihood that such large error regions will be connected to form a logical error that spans non-locally across the system [22]. The rate at which the system undergoes these phase transitions is denoted by the threshold for the system, given a particular error model. The threshold rate can greatly vary depending on the error model of the system, commonly spanning one or more orders of magnitude when comparing independent memory errors from circuit level noise (that is noise on the individual circuit components that are used to perform the error correction and detection).

4. Decoding qudit color codes

In this work, we use clustering techniques to decode the qudit color code. We will call this the GCC Decoder, for Generalized Color Clustering. In this description, we will focus on its application to the 6-6-6 code, but it can be just as easily applied to the 4-8-8 or 4-6-12 codes. We would like to point out that the goal of our decoding algorithm will be to find a correction process that will, along with the original error, return the state of the system to the codespace. The resulting correction therefore must combine with the error to result in no stabilizers being violated, or in the language of charge excitations, to cancel out all charge excitations through transport and fusion. In this section, we will discuss moving charges around by applying Pauli gates. These gates do not necessarily have to be applied physically, but rather can be tracked in software throughout further computations and it is this setting we have in mind when discussing these algorithms.

For plaquettes in qudit-valued topological codes to be stabilizers, we need to define an orientation on the code. For example, we can label the six data qudits surrounding each measurement qudit with numbers 0 through 5, increasing counterclockwise starting from the positive x axis. With this ordering in hand, we can assign positive signs to data qudits with even parity, and negative to data qudits with odd parity. This means that when we act with SUM gates on a measurement qudit—data qudit pair, we would add the charge of qudit 0, but subtract the value of qudit 1 from the total charge of the target measurement qudit. Figure 4 gives an example of such an orientation, represented in the dual picture.

In order for this to be a commuting stabilizer code, as explained thoroughly in the literature [23], we must choose one *privileged* color (usually red) to have the opposite sign convention as the other two, associating odd parity with positive. This is important to guarantee the commutativity of generalized X and Z plaquettes in the qudit code and can be tracked in software.

This orientation allows us to generate a valid and meaningful syndrome from our data errors on the color code. For example, if during our code cycle only a single data qudit error of magnitude k acted on our code, this would trigger the three surrounding measurement qudits in the following way: For B, R, and G, the data qudit is

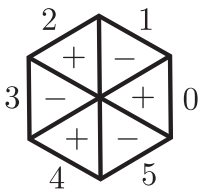

Figure 4. Single 6-6-6 code plaquette in the dual picture, with measurement qudits at vertices and data qudits at the center of each triangle. We define a counterclockwise orientation and an ordering of the data qudits with respect to the measurement qudit.

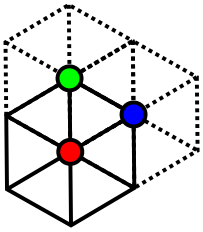

Figure 5. A given data qudit is at different position in the ordering for each of the surrounding measurement qudits.

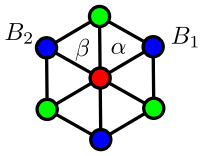

Figure 6. Red plaquette with data qudits α and β at positions 1 and 2. We can transport charge from B_1 to B_2 by modifying the values of α and β in the software.

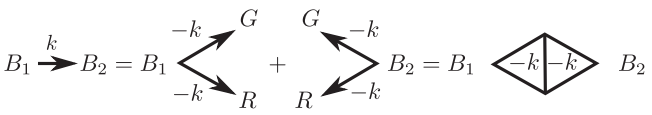

Figure 7. Transporting charge k from B_1 to B_2 is equivalent to the combination of transporting k from B_1 to R and G, and transporting $-k$ from B_2 to R and G.

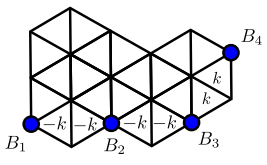

Figure 8. Illustration of charge transport from B_1 to B_4. Note that between B_3 and B_4 we add charge k to both data qudits because the parity is odd according to our specified orientation.

at (odd) positions 1, 3 and 5, so during the code cycle, each of these triggers $-k$, where $-k$ is the additive inverse of k modulo qudit dimension D. In general, a given data qudit will be at positions of the same parity for each of the neighboring measurement qudits. This process is summarized in figures 5–8.

Inverting this process tells us how to transport charge along the lattice. The simultaneous transport of charge k from R to the two other measurement qudits in the triangle is accomplished by changing the value of the data qudit enclosed by the triangle (which can be performed with post-processing software). In particular, we must subtract the signed charge k (modulo D) from the data's initial charge. The sign, as above, is determined by

the parity of the data qudit with respect to R. In this process, both B and G pick up a charge of $-k$ regardless of data qudit sign.

This leads us to the following key observation: by composing such transport processes, we can transfer charge between qudits of the same color without affecting measurement qudits of complementary types. When clustering in the surface code, we were able to annihilate a neutral cluster through controlled charge transport. This same-color transport allows us to perform a similar procedure for qudit color codes.

Here, adding k to qudits α and β transports charge k from B_1 to B_2. In this way, we can transport charge across greater distances on the lattice by composing these transportations.

4.1. GCC decoder

At this point we can introduce our decoder. Similarly to the surface code case, we will have a notion of *boundary-neutral*, however for the color code this will be slightly more nuanced. In the surface code, we called a cluster boundary-neutral if the cluster contained *any* boundary element. We proceeded to annihilate all boundary clusters before iterating the scale. For the color code, even if a cluster contains a boundary element, we can not necessarily annihilate the entire cluster. Therefore, we will refer to a cluster as boundary-neutral only when we can annihilate the cluster through internal charge transport and transport with the included boundaries.

The decoding proceeds as follows: we define our primitive unit of distance as the length of an edge on the dual lattice l. Starting with $i = 1$ (the scale factor), we connect all measurement qudits in the syndrome separated by distance $\leqslant l \times i$, and identify all maximally disjoint clusters. Within each cluster, we calculate the average position, and then transport the charge from all elements in the cluster toward the measurement qudits closest to that central point[10], making corresponding data qudit changes in the software as we go. If the cluster is neutral, it will have been completely annihilated. Otherwise, we are left with at most two qudits (of distinct colors) at the center of the cluster. Boundary-neutral clusters are then annihilated, and the scale is iterated to the next integer value. the procedure is repeated until the syndrome is empty.

To determine if a particular decoding was successful (under a code capacity error model), it is sufficient to check that the original error plus correction commutes with a single logical operator (any of the three boundaries of the triangular code). An explicit example of a successful GCC decoding is given in the appendix.

5. Threshold estimates for qudit color codes

The performance of the different codes and decoders are tested in the memory bit-flip error model, for a variety of qudit dimensions. We only study the case of bit-flip noise as the code is a CSS code, and our decoding techniques would address errors of bit and phase type independently. Considering one type of noise is standard in estimating the performance of CSS codes under code capacity noise and will give us a natural comparison point to previous studies of color code decoders [24]. The memory model assumes noisy qudit memories, with perfect syndrome extraction and correction (according to the chosen decoder), and is generally considered to be a good representative of the performance of families of error correcting codes. That is, good performance under this error model will typically translate into good performance under gate noise when using fault-tolerant measurement circuits, although the threshold will typically decrease by at least an order of magnitude. Unless otherwise specified, each logical error probability data point represents 30 000 trials.

Explicitly, the error model we are studying here, in the case of qubits, is the following applied to each data qubit of the code:

$$\mathcal{E}(\rho) = (1 - p)\rho + pX\rho X. \tag{2}$$

Namely, the error model consists of random bit-flip noise being applied with probability p.

For qudits, each qudit in the code can be affected by $D - 1$ unique non-trivial integer multiples of each generalized bit-flip X operator. As a result, the error model generalizes to:

$$\mathcal{E}(\rho) = (1 - p)\rho + \frac{p}{D - 1}\sum_{j=1}^{D-1} X^j \rho X^{-j}. \tag{3}$$

To put our decoding results in context, we first present results for the surface code using MWPM and clustering techniques, and then present qubit color code results using the surface projection decoder of Delfosse [24], which relies on min weight perfect matching having projected the initial color code syndrome information onto multiple copies of the surface code.

[10] We also tried checking the net charge of a cluster first and as in the surface code case, only transporting charge in neutral clusters. However, empirically our collapse method, which preserves distance between the centers of clusters, gave better results.

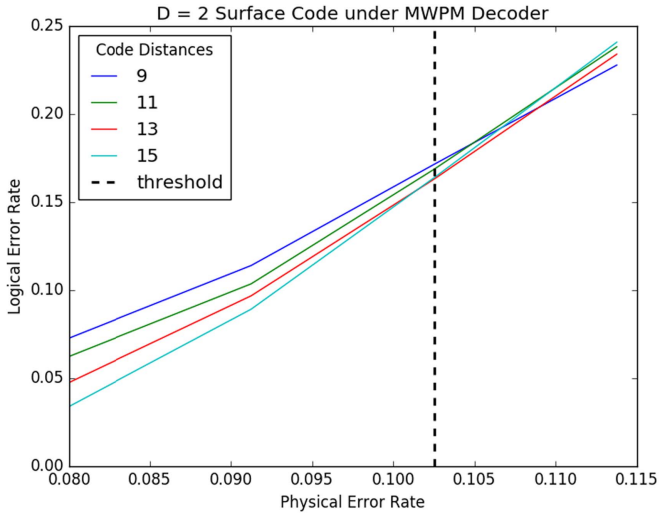

Figure 9. Logical error rates for the Kitaev surface code under min weight perfect matching decoder, which identifies maximally likely error chains. We have identified the threshold as the crossing of the distance 13 and 15 codes, at $p_{\text{thresh}} = 0.103$. Each data point corresponds to only 5000 trials, and trials were taken at fewer probabilities due to the relatively large computational cost of min weight matching as compared to renormalization techniques.

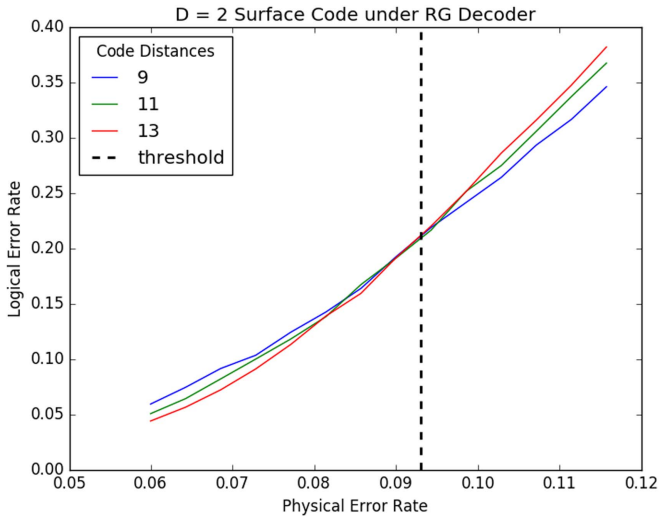

Figure 10. Logical error rates for Kitaev surface code under hard decision renormalization group decoder. As expected, in this qubit case, clustering slightly under-performs compared to the minimum weight perfect matching decoder, giving a threshold at $p_{\text{thresh}} = 0.093$.

5.1. Surface code threshold estimates

The first set of thresholds we present are for two different decoders in the surface code. While the MWPM decoder provides an optimal solution for the presence of excitations in the stabilizer dual graph caused by errors, its generalization to higher-dimensional systems does not scale well, as previously discussed. As such, in parallel we present the clustering algorithm which will be used at higher dimensions, and will take a hit in terms of threshold rate in the case of qubits ($D = 2$). Our simulation of the MWPM decoder aligns with previous theory and simulations, converging to a threshold value of 0.103, see figure 9.

In contrast, the clustering algorithm based on renormalization yields a threshold result of 0.093 in the case of qubits, see figure 10. However, this decoding algorithm can then be generalized to higher-dimensional qudit systems, as outlined in the previous sections. The shift in the threshold value obtained from the crossing point of the logical error rate at various distances can be explicitly seen as the qudit dimension increases, see figures 11 and 12. The threshold value increases monotonically as the qudit dimension rises, saturating close to a threshold value of 0.155 as shown in figure 13.

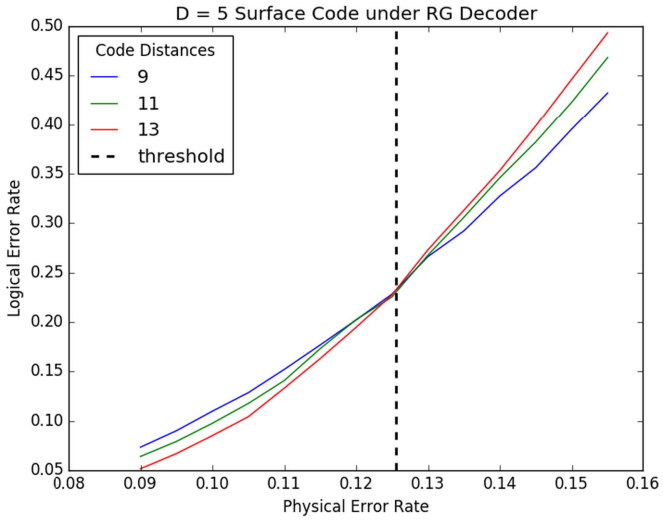

Figure 11. Kitaev surface code under hard decision renormalization group decoder for qudit $D = 5$. $p_{\text{thresh}} = 0.1255$.

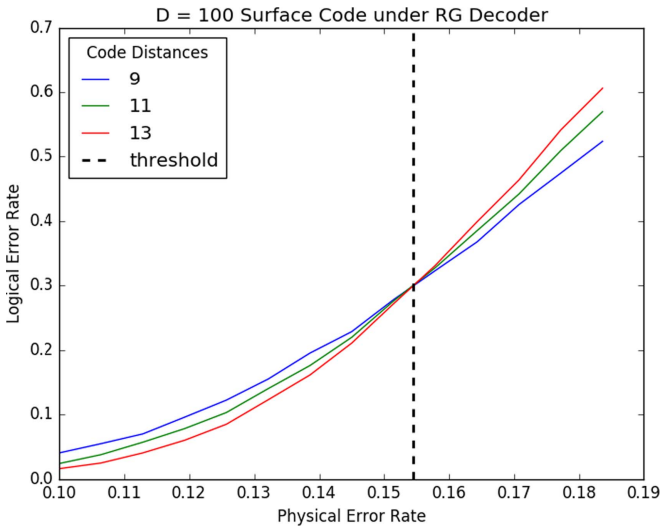

Figure 12. Kitaev surface code under hard decision renormalization group decoder for qudit $D = 100$. At $p_{\text{thresh}} = 0.1545$, we have almost reached a plateau.

5.2. Color code threshold estimates

Before describing decoding methods for qubit and qudit color codes, we caution against direct comparison of specific threshold values between the surface codes discussed above and 6-6-6 color codes, as they have different underlying geometries. The Delfosse surface projection (DSP) method for decoding the qubit color code is based on the idea of projecting two of the three colors onto a surface code, and then using a surface code decoder to obtain a set of different correction paths. Unifying the correction paths from these different projections determines a valid correction for the color code, at the expense of reducing the threshold value [24]. This decoding method achieves the highest known decoding rate in terms of code capacity noise for the color code. There are two potential methods then to generalize such a decoding technique to higher-dimensional qudits, either by using similar projections and a generalized decoder at the surface code level, such as the renormalization decoder analyzed in the surface code, or to generalize the clustering algorithm directly. We chose the latter, adapting the renormalization group decoder into a general color clustering (GCC) decoder.

The results for the color code decoders are presented in figures 14–18. The implementation of the DSP decoder realizes a threshold rate of 0.080 for code capacity noise in the qubit model, slightly below the result obtained in theory and simulation. This threshold, which applies only to *our* particular implementation differs from Delfosse's result of 0.087. Yet even the threshold for our DSP implementation exceeds the GCC decoder in

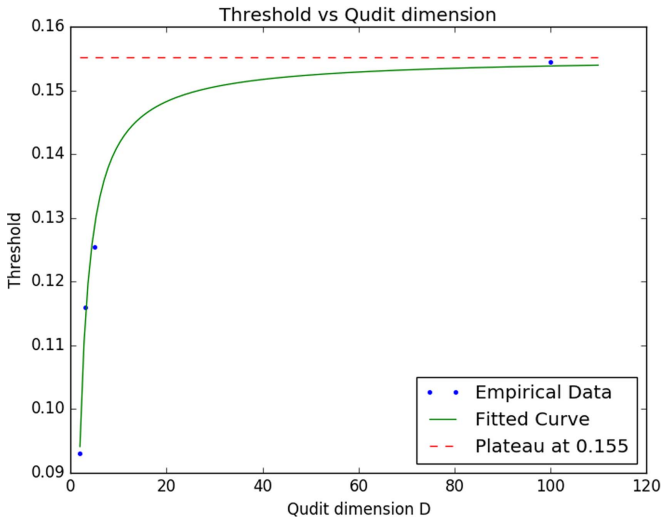

Figure 13. The threshold increases with qudit dimension until saturating around $p_{\text{thresh}} = 0.155$. Curve fitted to function $T(x) = T_{\text{plateau}} - \frac{\alpha}{\beta - D}$, where D is the qudit dimension, T is the threshold, and T_{plateau} is the threshold as $D \to \infty$.

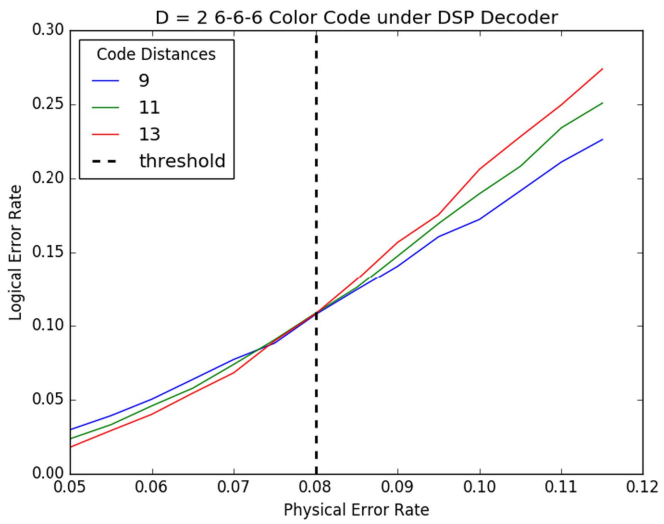

Figure 14. Threshold estimates using the surface projection decoder, which employs min weight perfect matching on each of the code's shrunk lattices. $p_{\text{thresh}} = 0.080$.

the case of qudits. Moreover, in the limit of large qudit dimension the thresholds obtained from the GCC decoder surpass that of the ideal qubit color code decoding.

The surface projection decoder for qubit color codes works by splitting a color code into three shrunk lattices, and performing MWPM on each lattice. Just like the hard-decision renormalization group algorithm for surface codes, our clustering algorithm approximates a hard problem by breaking up a global error syndrome into smaller, localized correction clusters. In analogy to the surface code case, then, we would expect our GCC decoder to attain a slightly lower qubit threshold than the DSP decoder. In addition, we should see threshold increasing with qudit dimension. In fact, we see just this: the clustering decoder exhibits a lower threshold for qubits, attaining a rate of 0.056, which is an even more substantial similar drop from ideal threshold value than in the surface code case. However, in generalizing this decoding technique to higher-dimensional systems, the threshold rate increases to 0.084 ($D = 3$), and to 0.115 ($D = 25$), eventually plateauing at 0.119, see figures 15–19. A summary of our results are presented in table 1.

Unlike in the surface code, in the color code of code distance 7, every internal measurement qudit is connected to at least one boundary from the first clustering iteration onward. Because of this, our decoder performs poorly. These *small-size* effects become less important for code distance 9, and vanish almost entirely for larger codes. In order to properly reflect these small-size effects, we do not consider this curve when establishing the threshold. However due to length of the classical simulation we were only able to simulate lattice

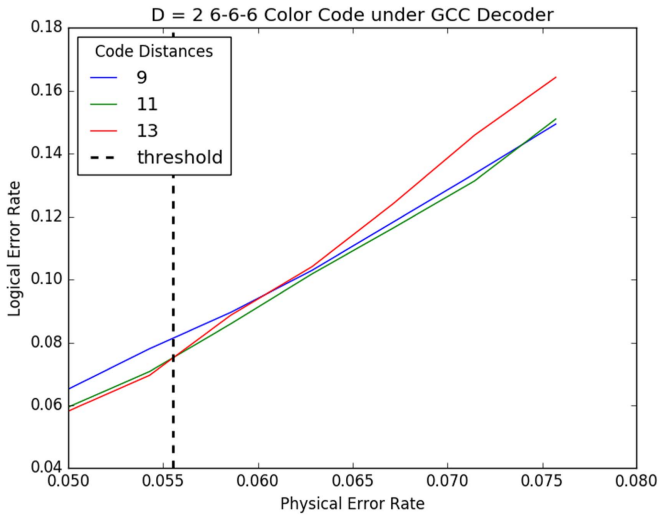

Figure 15. 6-6-6 color code under general color clustering decoder for qubit $D = 2$. $p_{\text{thresh}} = 0.056$.

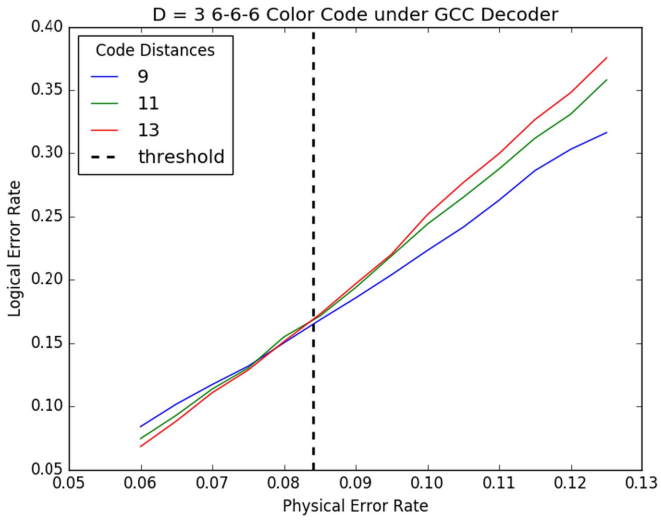

Figure 16. 6-6-6 color code under general color clustering decoder for qudit $D = 3$. $p_{\text{thresh}} = 0.084$.

sizes up to code distance 13. We used the crossing point of the code distance 11 and 13 as a point of reference for the threshold value and believe that further simulations at higher values would strengthen the assumption that the threshold value is at this point.

6. Conclusion

In this work, we have introduced a new decoding algorithm for qudit color codes to study the behavior of the fault-tolerance threshold as qudit dimensionality is increased. Since algorithms that use MWPM as a primitive, such as the DSP method [24], do not scale well with increased dimension, a new scheme based on renormalization group techniques is proposed called GCC. This mirrors similar work completed by Watson *et al* in the case of surface codes [11], and similar conclusions are drawn. Namely, the memory threshold drops in the case of qubits from 0.080 in the case of the surface projection method used in our simulations to 0.056 for the case of GCC. However, as qudit dimension is increased, the threshold rate rises to a saturation point of near double that of the qubit case, that is 0.119. We expect that the decoder would exhibit a similar behavior when generalizing the noise to include measurement errors, phenomenological noise, however leave this study for future works.

It should be noted that while the threshold value for qudit codes exceed that of the qubit base, a direct comparison may be misleading. Namely, while the probability of introducing an error is the same, the

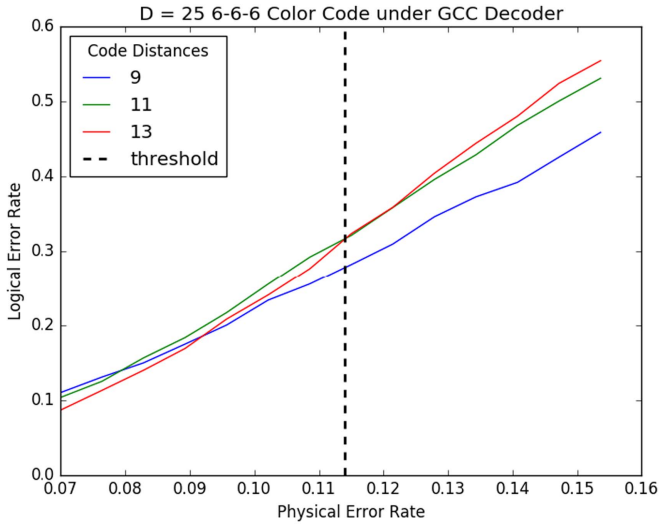

Figure 17. 6-6-6 color code under general color clustering decoder for qudit $D = 25$. $p_{\text{thresh}} = 0.115$.

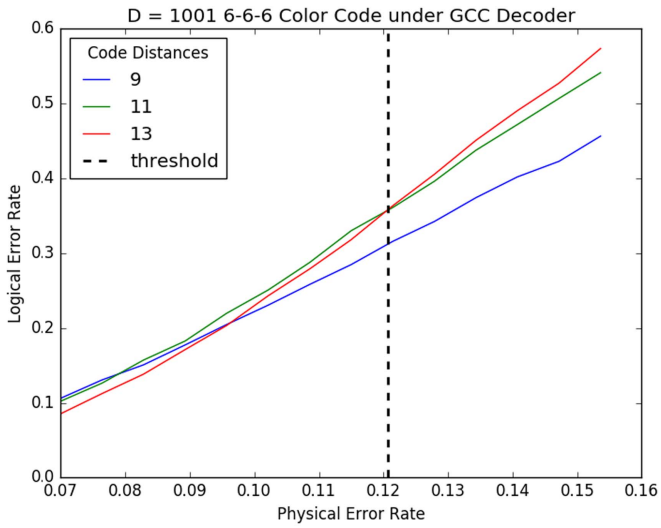

Figure 18. 6-6-6 color code under general color clustering decoder for qudit $D = 1001$. $p_{\text{thresh}} = 0.1207$.

probability of a given error type is reduced in the qudit case since there are an increased number of possible errors due to the growth in system size. One would expect any experimental implementation of qudit codes to be hampered by the increased dimensionality of the system size in terms of potential error leakage, as there are increased degrees of freedom that can be coupled to the environment. Whether such noise can be reduced to sufficiently small levels in order to take advantage of the properties that qudit codes provide remains an interesting question for experimental implementations.

For the qudit surface code, the effect of adding initialization steps to their clustering decoder to account for path degeneracies has additionally been studied. It would be interesting to perform a similar analysis with our qudit color code decoder, as in the 4-8-8 and 4-6-12 codes certain error chains are exponentially suppressed.

Going forward, we would also like to generalize our clustering algorithm to 3D qudit color codes, and to explore possible uses of clustering in qudit gauge color codes.

Acknowledgments

The authors thank Michele Mosca for fruitful discussions and comments regarding the manuscript, and the IQC technical staff for providing access to the Heavylift computing clusters at the IQC. We also thank Earl Campbell and Benjamin Brown for useful comments. JM acknowledges support from a Yale Richter Fellowship and

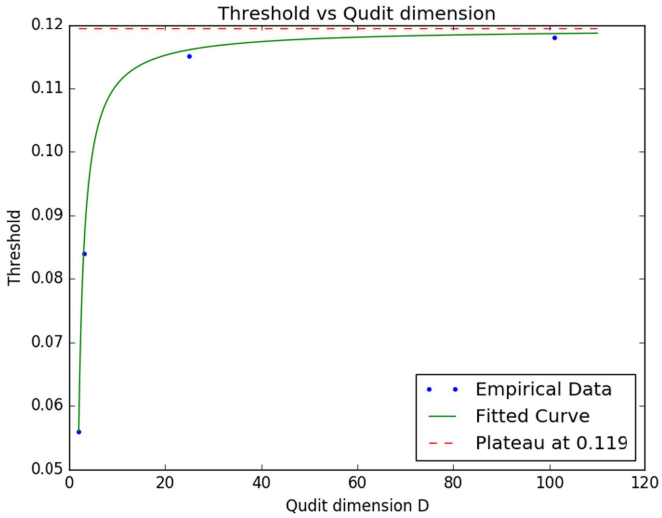

Figure 19. As in the surface code case, the threshold increases with qudit dimension until saturating. The plateau occurs at $p_{\text{thresh}} = 0.119$. Curve fitted to same form as in figure 13, including the data point at $D = 1001$ not shown in this plot.

Table 1. Code capacity threshold results for the implemented simulations of the surface and (6.6.6) color codes. In the case of the surface code, the optimal min-weight perfect matching (MWPM) is compared to the hard-decision renormalization group (HDRG) decoder for qubits ($D = 2$), and the latter is then generalized to higher qudit dimension. Similarly, for the color code, the Delfosse surface projection (DSP) method is compared to the general color clustering (GCC) decoders and further generalized.

Code	Decoder	$D = 2$	$D \to \infty$
Surface Code	MWPM	0.103	
	HDRG	0.093	0.155
(6.6.6) Color Code	DSP	0.080	
	GCC	0.056	0.119

Undergraduate Research Award from the University of Waterloo. TJO acknowledges support from NSERC through the Vanier CGS as well from the Burke Institute through the Sherman Fairchild Fellowship. VG acknowledges support from NSERC and CIFAR. IQC is supported in part by the Government of Canada and the Province of Ontario.

Appendix. GCC decoder

In this appendix, we provide some additional detail on the GCC used to decode qudit color codes. As mentioned in the main text, the decoding algorithm takes as inputs the qudit stabilizers that have been violated, and uses techniques from renormalization group decoders to cluster sets of corrections. Figure A1 provides a graphical representation of the dual lattice of the 6-6-6 color code with distance 11, where qudit stabilizers measure the data errors according to the orientation described in section 4.

A set of qudit errors triggers the appropriate set of measurement flags which will serve as input to the decoder. Figure A2 provides an example of the stabilizer measurements observed in a qudit color code with dimension 3. Since the GCC decoder acts as a CSS decoder, addressing errors of X^k and Z^k type separately, we will assume for simplicity that all of the errors are of the same type, say X. The qudit errors are represented by gray circles, where a single line contour represents a X^1 error, while a double contour represents a $X^2 \cong X^{-1}$ error. Only the violated stabilizers are shown in the figure, and they register the appropriate charge according to the orientation of the measured errors. In the example, the error in the bottom right is of type X^1, and since it is in the positive orientation of the neighboring stabilizers, they register violations corresponding to charge 1.

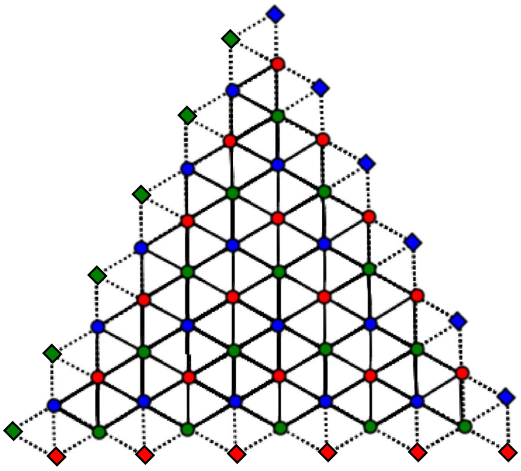

Figure A1. The dual lattice for the 6-6-6 color code with distance 11. The vertices with circles represent stabilizers of a given color, while those with diamonds represent the appropriate boundary. Data qudits reside at plaquettes, therefore each stabilizer measures 6 data qudits (or less at the boundary).

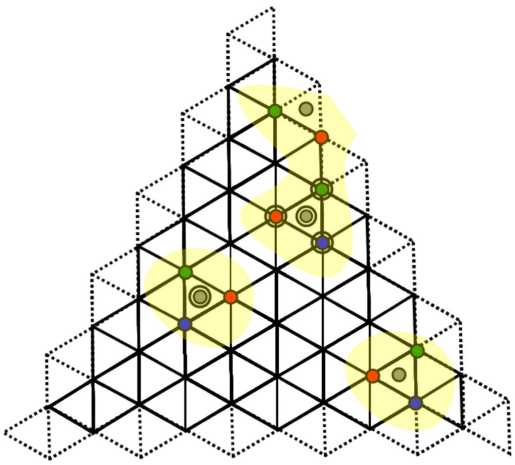

Figure A2. Example of a set of errors for the qudit color code of dimension 3. Errors are given in gray, and are of a single type, say X. The errors of charge 1, that is X^1, are represented by circles with a single contour line. Errors of charge 2, that is X^2, are represented by double contour lines. Only violated stabilizers are shown in this graph, and their charge is represented as in the case of the qudit errors.

However, in the error of charge 2 on the left, the neighboring stabilizers register the error as of charge (-2), therefore equivalent to charge 1.

The clustering algorithm takes all input stabilizers and their associated graph, and clusters all violated stabilizers that are within distance 1 of each other (that is, each cluster must consist of violated stabilizers that are at least of distance 2 from any other cluster). All charges within the cluster are then brought together to three geometrically central nodes, that is one for each color. This is done through charge transport. In our algorithm, we reduce the three central charges to at most two central charges by canceling out at least one of the charges through charge identities between the three colors. If a given cluster is charge neutral, then all charges are removed and the correction is complete (for that cluster). Any remaining charge is then iteratively fed back into the same algorithm, where clusters are now formed by joining charges that are distance 2 apart. The process continues until all charges are removed. In figure A2, the yellow regions represent the different clusters.

Figure A3 shows the remaining stabilizer violations after the first step of the GCC algorithm for the discussed example. The introduced correction, due to charge transport, can combine from different colors to cancel, and the remaining error (initial error plus correction) are given in gray. Therefore, since the top-right cluster from figure A2 is not charge neutral, there remains a charge after the initial correction, represented by the blue violation in figure A3. At the second step, the charge is close enough (distance 2) to the Blue boundary to cancel out the charge with the boundary, therefore correcting the error locally in the top-right section of the graph. As such, the decoder will successfully correct this form of error. However, it should be noted that the

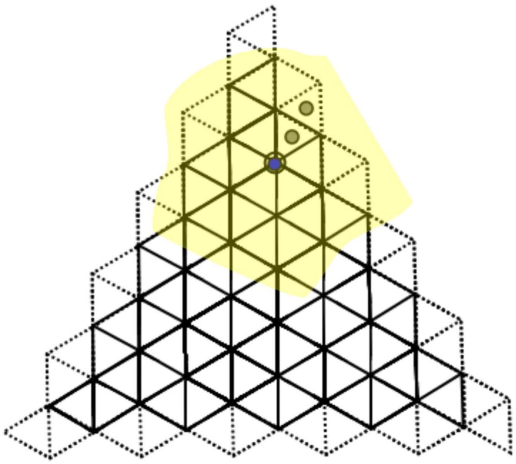

Figure A3. The remaining violated syndromes (in software) after the proposed charge transport corrections from the first step of the GCC decoder on the errors from figure A2. The remaining combination of initial errors and introduced errors are given in gray.

growth in the cluster size can very quickly span across the graph, resulting in potential logical error, if there are too many initial non-neutral clusters. This will happen when close to threshold, due to percolation, and as such the threshold of such a scheme is lower than ideal decoders such as that proposed by Delfosse [24].

References

[1] Shor P W 1995 Scheme for reducing decoherence in quantum computer memory *Phys. Rev.* A **52** R2493–6
[2] Steane A M 1996 Multiple-particle interference and quantum error correction *Proc. R. Soc.* A **452** 2551–77
[3] Calderbank A R and Shor P W 1996 Good quantum error-correcting codes exist *Phys. Rev.* A **54** 1098–105
[4] Gottesman D Stabilizer codes and quantum error correction http://arxiv.org/abs/quant-ph/9705052
[5] Briegel H J and Raussendorf R 2001 Persistent entanglement in arrays of interacting particles *Phys. Rev. Lett.* **86** 910–3
[6] Aharonov D and Ben-Or M 1997 Fault-tolerant quantum computation with constant error *Proc. 29th Annual ACM Symp. on Theory of Computing, STOC '97* pp 176–88 New York, NY: ACM
[7] Gottesman D 2000 Fault-tolerant quantum computation with local gates *J. Mod. Opt.* **47** 333–45
[8] Raussendorf R and Harrington J 2007 Fault-tolerant quantum computation with high threshold in two dimensions *Phys. Rev. Lett.* **98** 190504
[9] Kitaev A Y 2003 Fault-tolerant quantum computation by anyons *Ann. Phys., NY* **303** 2–30
[10] Bombín H and Martin-Delgado M A 2006 Topological quantum distillation *Phys. Rev. Lett.* **97** 180501
[11] Watson F H E, Anwar H and Browne D E 2015 Fast fault-tolerant decoder for qubit and qudit surface codes *Phys. Rev.* A **92** 032309
[12] Duclos-Cianci Guillaume and Poulin David 2013 Kitaev's d-code threshold estimates *Phys. Rev.* A **87** 062338
[13] Anwar H, Brown B J, Campbell E T and Browne D E 2014 Fast decoders for qudit topological codes *New J. Phys.* **16** 063038
[14] Marks J 2016-2017 QTop https://github.com/jacobmarks/QTop/
[15] Kubica A and Beverland M E 2015 Universal transversal gates with color codes: a simplified approach *Phys. Rev.* A **91** 032330
[16] Landahl A J, Anderson J T and Rice P R 2011 Fault-tolerant quantum computing with color codes arXiv:1108.5738
[17] Paetznick A and Reichardt B W 2013 Universal fault-tolerant quantum computation with only transversal gates and error correction *Phys. Rev. Lett.* **111** 090505
[18] Jochym-O'Connor T and Laflamme R 2014 Using concatenated quantum codes for universal fault-tolerant quantum gates *Phys. Rev. Lett.* **112** 010505
[19] Bombín H 2015 Gauge color codes: optimal transversal gates and gauge fixing in topological stabilizer codes *New J. Phys.* **17** 083002
[20] Bombín H 2015 Single-shot fault-tolerant quantum error correction *Phys. Rev.* X **5** 031043
[21] Edmonds J 1965 Paths, trees, and flowers *Can. J. Math.* **17** 449
[22] Dennis E, Kitaev A, Landahl A and Preskill J 2002 Topological quantum memory *J. Math. Phys.* **43** 4452–505
[23] Brell C G 2015 Generalized color codes supporting non-abelian anyons *Phys. Rev.* A **91** 042333
[24] Delfosse N 2014 Decoding color codes by projection onto surface codes *Phys. Rev.* A **89** 012317